IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application. No: 10/676,634	§	Examiner:	Augustine, Nicholas
Filed: October 1, 2003	§		
Inventor(s):	§	Group/Art Unit:	2179
Luis M. Gomes, Madras S. Mohanasundaram	§	Atty. Dkt. No:	5150-82801
	§		
Title: EDITABLE DATA TOOLTIPS	§		
	§		
	§		
	§		

APPEAL BRIEF

Box: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

Further to the Notice of Appeal filed May 4, 2009, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1 and 6-23 are pending in the case. Claims 1-34 stand rejected under 35 U.S.C. § 102(b) and are the subject of this appeal. A copy of claims 1-34, incorporating entered amendments, as on appeal, is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been filed subsequent to the rejection in the Final Office Action of April 10, 2009. The Claims Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The present invention relates to the field of graphical programming, and more particularly to editable data tooltips for debugging programs.

Independent claim 1 recites a computer accessible memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, where during execution of the program the program instructions are executable by a processor to perform the following: Source code for the program is displayed on a display during execution of the program, where the executing program was compiled from the source code. *See, e.g., p.12:4-16; p.23:11-14; Figure 5:502.* First user input is received hovering a mouse cursor over an expression in the source code during execution of the program. *See, e.g., p.23:16-21; Figure 5:504.* In response to hovering the mouse cursor over the expression, a GUI element is automatically displayed proximate to the expression, where the GUI element includes a value of the expression. *See, e.g., p.24:2-p.25:2; Figure 5:506.* Second user input is received to the GUI element modifying the displayed value, thereby specifying a new value for the expression. *See, e.g., p.25:24-27; Figure 5:508.* The expression in the program is set to the new value in response to the second user input, where the program continues execution in accordance with the new value of the expression. *See, e.g., p.26:1-6; Figure 5:510, also, p.27:12-23, and p.18:1-6.*

Independent claim 18 is directed to a method for debugging a program during execution, and is a method analogue of claim 1. Source code for the program is displayed on a display during execution of the program, where the executing program was compiled from the source code. *See, e.g., p.23:11-14; Figure 5:502.* First user input is received hovering a mouse cursor over an expression in the source code during execution of the program. *See, e.g., p.23:16-21; Figure 5:504.* In response to hovering the mouse cursor over the expression, a GUI element is automatically displayed proximate to the expression, where the GUI element includes a value of the expression. *See, e.g., p.24:2-p.25:2; Figure 5:506.* Second user input is received to the GUI element

modifying the displayed value, thereby specifying a new value for the expression. *See, e.g., p.25:24-27; Figure 5:508.* The expression in the program is set to the new value in response to the second user input, where the program continues execution in accordance with the new value of the expression. *See, e.g., p.26:1-6; Figure 5:510, also, p.18:1-6.*

Independent claim 19 is directed to a system for debugging a program during execution, and is a system analogue of claims 1 and 18. The system includes a processor and memory medium coupled to the processor. *See, e.g., p.18:10-26; Figure 2A.* The memory medium stores program instructions executable by the processor to perform the following: Source code for the program is displayed on a display during execution of the program, where the executing program was compiled from the source code. *See, e.g., p.23:11-14; Figure 5:502.* First user input is received hovering a mouse cursor over an expression in the source code during execution of the program. *See, e.g., p.23:16-21; Figure 5:504.* In response to hovering the mouse cursor over the expression, a GUI element is automatically displayed proximate to the expression, where the GUI element includes a value of the expression. *See, e.g., p.24:2-p.25:2; Figure 5:506.* Second user input is received to the GUI element modifying the displayed value, thereby specifying a new value for the expression. *See, e.g., p.25:24-27; Figure 5:508.* The expression in the program is set to the new value in response to the second user input, where the program continues execution in accordance with the new value of the expression. *See, e.g., p.26:1-6; Figure 5:510, also, p.27:12-23, and p.18:1-6.*

Independent claim 20 is directed to a system for debugging a program during execution, and is a means plus function analogue of claims 1, 18, and 19.
The system includes means for displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code.
See, e.g., Figures 2A, 2B, 4, 6A-6C;
p.18, line 10-p.19, line 19,
p.21, line 21–p.22, line 25,
p.23, lines 8-14, and

p.28, line 32 – p.30, line 2.

The system also includes means for receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program.

See, e.g., Figures 2A, 2B, 4, 6A-6C;

p.18, line 10-p.19, line 19,

p.21, line 21–p.22, line 25,

p.23, lines 16-20, and

p.28, line 32 – p.30, line 2.

The system also includes means for automatically displaying a value of the expression in a GUI element proximate to the expression in response to said hovering the mouse cursor over the expression.

See, e.g., Figures 2A, 2B, 4, 6A-6C;

p.18, line 10-p.19, line 19,

p.21, line 21–p.22, line 25,

p.24, lines 2-23, and

p.28, line 32 – p.30, line 2.

The system also includes means for receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression.

See, e.g., Figures 2A, 2B, 4, 6A-6C;

p.18, line 10-p.19, line 19,

p.21, line 21–p.22, line 25,

p.25, lines 24-27, and

p.28, line 32 – p.30, line 2.

The system also includes means for setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression.

See, e.g., Figures 2A, 2B, 4, 6A-6C;

p.18, line 10-p.19, line 19,

p.21, line 21–p.22, line 25,

p.26, lines 1-14, and

p.28, line 32 – p.30, line 2.

Independent claim 21 recites a computer accessible memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, where during execution of the program the program instructions are executable by a processor to perform the following: Source code for the program is displayed on a display during execution of the program, where the executing program was compiled from the source code. *See, e.g., p.12:4-16; p.23:11-14; Figure 5:502.* First user input is received hovering a mouse cursor over an expression in the source code during execution of the program. *See, e.g., p.23:16-21; Figure 5:504.* In response to hovering the mouse cursor over the expression, the value of the expression is automatically displayed in a window proximate to the expression, wherein the window is operable to display a value of the indicated expression, wherein the window does not include window title bars or menus. *See, e.g., p.24:2-p.25:2; Figure 5:506.* Second user input is received to the window element modifying the displayed value, thereby specifying a new value for the expression. *See, e.g., p.25:24-27; Figure 5:508.* The expression in the program is set to the new value in response to the second user input, where the program continues execution in accordance with the new value of the expression. *See, e.g., p.26:1-6; Figure 5:510, also, p.27:12-23, and p.18:1-6.*

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1 and 6-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over Deutscher et al (US Pat. Pub. 2004/0001106, "Deutscher"), in view of Hampapuram et al. (US Pat. Pub. 2004/0221262, "Hampapuram").

VII. ARGUMENT

First Ground of Rejection

Claims 1 and 6-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over Deutscher et al (US Pat. Pub. 2004/0001106, "Deutscher"), in view of Hampapuram et al. (US Pat. Pub. 2004/0221262, "Hampapuram"). Appellant respectfully traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 10, 11, 12, 17, 18, 19, 20, 21

Independent claim 1 is separately patentable because the cited reference does not teach or suggest the limitations recited in this claim. For example, Appellant respectfully submits that the cited art fails to disclose, **displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code**, as recited in claim 1.

Deutscher is directed to a multimedia presentation production system in which presentation data are separated from presentation logic (see, e.g., [0008]). Per [0009], Deutscher discloses a tool consisting of "graphic user interfaces for easily entering the data associated with the rich media presentation", where "the tool is also designed to make it simple and easy to edit the data of the underlying schema." Per [0017]-[0018], among the various data entry graphical user interfaces (GUIs) disclosed are "data entry grids", which are displayed in a presentation tool window, and "allow the user to enter the information into the presentation data file and the master track media file that is needed to drive the timeline of the presentation. The first of these grids, which is displayed by default once the master track program has been imported, is the scripts grid. The scripts grid is where the user enters events called script commands that will be triggered during the playback of the presentation."

The Office Action incorrectly equates Deutscher's scripts grid with Appellant's claimed source code of an executing program. Per Deutscher, and as described in more detail in section 2.6.1 ([0137] – [0144]), the scripts grid is "**essentially a scheduling table** where the user enters events (sometimes referred to as script commands) that will be triggered during the playback of the presentation." The entered data in the various data grids (including the scripts grid) are saved to the presentation *data* file, which is then referenced by the presentation system to present the

multimedia presentation. A scheduling table is *not* source code that is compilable to generate an executable program, but rather is a means for specifying *input data*.

In direct contrast, as one of skill in the programming arts would readily understand, "source code" is a term of art in the programming domain and refers specifically to program instructions in a programming language that are compilable to produce executable code that may be run on a processor. This is emphasized in the independent claims by the limitation: "**wherein the executing program was compiled from the source code**", which is certainly not the case with the cited script grid. Deutscher does not describe any of the data grids as source code for an executing program. Per cited paragraph 137, the program that is paused to allow for Deutscher's scripts grid editing is "a video or audio program that is designated as the master track", *not* an executable program for which the scripts grid is source code. In the Response to Arguments, the Office Action continues to assert that Deutscher's script grid is equivalent to the source code of claim 1, arguing that "each perform [*sic*] the same functionality". This is not a proper analysis or characterization of the technical content of claim 1. Appellant respectfully reminds the Examiner that *every word in the claim* is to be given consideration, and that the terms of the claim are to be interpreted in light of the Specification. Terms of art should not be re-interpreted contrary to the Specification. A scheduling table (Deutscher's description of the script grid) used by an executing program to manage multi-media presentations does not "perform the same function" as source code that is compiled to generate the executable program. Note that substituting either of these elements for the other in their respective inventions would render the respective inventions inoperable, and so they clearly do *not* perform the same function, and are not equivalent. One of ordinary skill in the programming arts would readily understand that an editing tool and a data file used by an executing program (the presentation viewer) are *not* source code for the executable program, **wherein the executing program was compiled from the source code**, as recited in claim 1. Per Deutscher, the script grid is a data entry grid that allows the user to enter information (i.e., *data*) into the presentation data file, and the presentation data file is (obviously) a *data file*, which is used by the presentation viewer (the executable program) to present a presentation. Neither the script grid, nor the data produced by the script grid, nor the presentation data file, is compiled into executable code, i.e., executable program instructions, contrary to the Examiner's assertions.

The Examiner further argues that in Deutscher the user can edit the script bar during execution of the program, citing paragraphs [0137], [0155], [0200], and [0201] as well as Figures 13, 17, and 26B. Figure 26B does not appear to be germane to this feature. Paragraphs [0200] and [0201] related to data entry regarding presentation material ordering, and a presentation

"build" process. However, Appellant notes that Deutscher's presentation system is data driven, whereby a presentation viewer processes formatted data (from the presentation file) to present multimedia content. For example, Appellant notes that Section 6.0 ([0202]-[0208]) states the following (in [0203]): "The previously mentioned project builder module of the presentation production system takes all the data input by the user via the user interfaces and builds a presentation package that can be run by the presentation viewer." No program compilation (as understood in the programming arts) is performed to generate an executable program. Moreover, this test clearly states that the cited grids are means for inputting data, not compilable source code.

Paragraph [0137], analyzed above, clearly indicates that the scripts grid is a means for providing input data for the presentation viewer (in the form of a presentation data file), *not* source code that can be compiled into executable program instructions. Nor does this citation mention or even hint at the user editing these data *during execution of the presentation viewer*; nor is such data source code for the program (the presentation viewer). Figure 13 shows a text entry dialog box whereby the user can overwrite textual data for the script grid data table, but, as noted previously, the data do not include source code that is compilable to executable program instructions of the presentation viewer. Paragraph [0155] and Figure 17 are directed to editing transcription entries via a text entry dialog box, wherein the user selects a data field, e.g., a time code field, and overwrites the data therein. Again, these citations in no way teach that the transcription entries or the transcription grid in general comprise source code that is compiled to generate the executable program instructions of the presentation viewer, nor that the transcription entries are edited during execution of the presentation viewer. Moreover, in paragraph [0152], Deutscher states that the default language for the transcriptions is *English*, which is decidedly *not* a programming language; the transcription is human-readable text, not program source code. The contents of the grids are data, not source code that is compilable to executable program instructions. As Appellant has explained repeatedly, the transcription text field shown in Figure 17 includes what appears to be a FOR loop designation, but this is actually ***text of a tutorial regarding programming in OLE and Visual Basic***, as may be seen by examining the text of the transcription grid visible behind the data entry dialog box. Thus, the "FOR loop", which Appellant notes includes generic "<code block>" designations, as well, is a textual passage that would be displayed to the user of the presentation viewer while the corresponding audio track of the tutorial is being presented audibly. The cited text is not intended for compilation, nor, Appellant notes, is it compilable at all, as one of skill in the programming arts can readily see. The Examiner continues to argue that this tutorial text is source code for the executing program,

which is incorrect. The Examiner appears to assert this (incorrect) equivalence based on certain asserted high-level similarities between the two items. However, this is not the proper way to analyze claim limitations. Note that if one considers any two items from a high enough level of abstraction, *everything* is equivalent, e.g., a porkchop and an apple are both edible objects, but are not equivalent. It is improper for the Examiner to disregard or redefine technical terms in Appellant's claims, thereby refusing to give patentable weight to these terms. Appellant requests that the Examiner give proper weight and meaning to the limitation "displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code", and submits that if such standard interpretation of these well-known technical terms is used, the Examiner's asserted equivalence will clearly be seen to be incorrect. Hampapuram also fails to teach this claimed feature. Appellant thus submits that the cited art fails to disclose this feature of claim 1.

Nor does the cited art disclose **receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;** nor **in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression**, as recited in claim 1.

Hampapuram and Deutscher fail to disclose the claimed hover invoked functionality. As discussed above, Deutscher discloses editing a script grid, which, as Deutscher defines it, is "essentially a schedule table", which is not source code as defined in claim 1. Paragraphs [0200] and [0201], discussed above, do not support the Office Action's assertion that the grids are source code compilable to generate an executable program, and, as mentioned above, [0203] makes it clear that the grids are means for inputting data for the presentation viewer: "The previously mentioned project builder module of the presentation production system takes all the data input by the user via the user interfaces and builds a presentation package that can be run by the presentation viewer." Nor is Deutscher's editing of the script grid performed during execution of the program. Appellant further notes that Deutscher teaches displaying the pop-up window in response to user input, specifically, double-clicking on the item to be edited, whereas in claim 1, the GUI element is automatically displayed in response to simply hovering the mouse cursor over the expression. Cited [0180] discloses hovering over a script element, e.g., a script command icon to invoke display of information related to the element, but not an expression. Similarly, cited [0183] is directed to editing data in a data grid, and is not germane to this claimed feature. Nor are Deutscher's icons, events, and so forth, expressions in program source code, as would be

readily understood by one of skill in the programming arts. Moreover, in asserting that Deutscher discloses these features, the Office Action cites Deutscher's double-clicking on icons, event labels, markers, etc., in the data grids (Figure 13, 24A, [0180], and Figures 23-24C), arguing that hovering could also be used to invoke a pop-up window; however, since Deutscher's data grids are not program source code, and Deutscher's data grid icons, event labels, event lines, markers, etc. are not expressions in program source code, this point is irrevelevant. Nor are Deutscher's pop-up menus tooltips.

Nor does Hampapuram remedy these deficiencies of Deutscher. For example, per Hampapuram's Abstract, the macro expansions are processed by Hampapuram's tool during the build process of a project ("during a build of a programming project's source files"), where this processing operates to collect and record the macro expansion information into an output file or database. The tool then uses the recorded information to "display the macro expansions in a graphical user interface of the tool, such as for source browsing or viewing static analysis". Appellant notes that "source browsing" and "viewing static analysis" are not run-time operations, and are *nowhere described as being performed while the program is executing*. "Expanding a macro" is not at all the same as automatically displaying a GUI element proximate to the expression (over which the user hovers a mouse cursor *during execution of the program*), where the GUI element includes a value of the expression. More particularly, Hampapuram's GUI does not display the value of an expression during execution of the program, but rather simply displays a macro expansion statically, i.e., *not* at runtime. Appellant has studied cited [0007], [0020], [0022] and Figure 3 carefully, and respectfully submits that these citations are clearly directed to static or edit/compile time expansion of macros, not at runtime, and thus are not germane to these features. Nowhere does Hampapuram describe the macro expansions occurring at runtime. Similarly, cited paragraph [0009] is similarly directed to static (edit or compile time) expansion of macros, and is similarly irrelevant to these claimed features.

The Office Action's citation of Deutcher's Figure 11 and paragraphs [0137]-[0140] is also not germane, at least because the development processes disclosed is not with respect to code development, but rather, presentation data development.

Thus, the cited art fails to teach or suggest these features of claim 1.

Nor does the cited art disclose **receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and setting the expression in the program to the new value in response to the second user input,**

wherein the program continues execution in accordance with the new value of the expression, as recited in claim 1.

Deutscher and Hampapuram fail to disclose displaying a value of an expression *in source code of an executing program*, i.e., at runtime, nor modifying the value of such an expression in the source code of an executing program, *where the program continues execution in accordance with the new value of the expression*. [0143] of Deutscher is directed to editing of a data grid, specifically, a scripts grid, which, as explained above, is not compilable source code for an executing program, but rather is a means for specifying input data for a presentation viewer. Deutscher's editing of the script grid neither modifies an expression in source code as claimed, nor is performed during execution of the program. Nor do Figures 13 and 17, described above, disclose these features. In fact, Deutscher is in general not germane to Appellant's claimed invention at all.

Similarly, Hampapuram's macro expansions do not teach or suggest modifying the value of an expression in the source code of an executing program, where the program continues execution in accordance with the new value of the expression, and are not performed during execution of the program. As noted above, cited [0007], [0020], [0022], [0009], and Figure 3 disclose static or edit/compile time expansion of macros, not at runtime, and thus are not germane to these features. Nowhere does Hampapuram describe the macro expansions occurring at runtime. Similarly, the Office Action's citation of Deutcher's Figure 11 and paragraphs [0137]-[0140] is also not germane, at least because the development processes disclosed is not with respect to code development, particularly dynamic debugging, but rather, static presentation data development.

The Office Action's suggested motivation to combine is incorrect and improper. The Office Action asserts that it would have been obvious to one of ordinary skill in the art to combine Hampapuram and Deutscher because "one of ordinary skill in the art would recognize the program being used in the system of Deutscher does not have to be program specific for the functionality of a pop-up control and that the pop-up control could work in any program environment (e.g., a debugger)", and further asserts that "the combination of Hampapuram into Deutscher would yield the predictable result of having a control pop-up window which is initiated by hovering with the mouse courser over an area of interest by the user in such that the user is able to input data into the pop-up window upon presentation of pop-up window by the system [*sic*]".

The Examiner characterizes the functionality taught by a combination of Hampapuram and Deutscher as "having a control pop-up window which is initiated by hovering with the mouse courser over an area of interest by the user in such that the user is able to input data into the pop-up window upon presentation of pop-up window by the system". However, Appellant respectfully notes that there is a substantial difference between simply changing a value in a static editor as taught by Deutscher or expanding a macro in a source code browser or viewer as taught by Hampapuram, and dynamically modifying a value of an expression during execution of a program as claimed, i.e., via user input to a tooltip invoked via hovering the mouse over the expression during runtime, as claimed. For example, note that Deutscher edits text in the script grid, e.g, script type, script parameter, which is subsequently referenced by the program upon execution when the program accesses the presentation file to which the script grid's changes have been copied. This is not a dynamic process performed at runtime, and is very different from Appellant's dynamic modification of a value of an expression during runtime, where the executing program continues to execute using the modified value.

Nor does Hampapuram's GUI facilitate editing the value of an expression *during execution of the program*, but rather simply displays a macro expansion statically, i.e., *not* at runtime, as explained in detail above. As also discussed, Deutcher's Figure 11 and paragraphs [0137]-[0140] are also not germane, at least because the development processes disclosed is not with respect to code development, particularly dynamic debugging, but rather, static presentation data development.

Appellant respectfully submits that since neither reference discloses these features, one of skill in the art would not be compelled to combine them in an attempt to generate Appellant's invention, and so Appellant submits that a proper motivation to combine has not been provided, and thus, Hampapuram and Deutscher are not properly combinable to make a prima facie case of obviousness.

Moreover, even were Hampapuram and Deutscher properly combinable, which Appellant argues they are not, the resulting combination would still not produce Appellant's invention as claimed, as explained at length above.

Thus, for at least the reasons provided above, Appellant submits that the cited art of Deutscher and Hampapuram, taken singly or in combination, fails to teach or suggest all the features and limitations of claim 1, and so claim 1, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 6, 7, 8

In addition to the novel limitations of claim 1, the cited art fails to teach or suggest **wherein the GUI element is context sensitive**, as recited in claim 6.

Cited Figure 17 shows a view of part of a work sector of a user interface screen with a portion of the transcription grid and a pop-up transcription entry editing window. More specifically, as the relevant text in paragraph [0155] describes, a transcription entry pop-up window appears with the transcription text and time code of the selected entry appearing in the appropriate places in the window, i.e., the time code appears in the time code field, and the transcription text appears in the transcription text field, where the user can change the transcription entry data by selecting and overtyping the existing data. Clearly, this figure disclose straightforward editing of properties via specific edit fields for the particular grid being edited (in this case, the markers grid), and in no way discloses a context sensitive GUI element as defined in the claim, specifically, where the GUI element is displayed in response to hovering the mouse cursor over an expression in source code during execution of the program (generated via compilation of the source code). Nor does Deutscher ever describe or otherwise make mention of any context sensitive GUI.

Regarding various dependent claims (from claim 6) the Office Action cites Figures 13 and 17. However, Figure 13 shows specific edit fields for editing script attributes in a scripts grid, and Figure 17 shows specific edit fields for editing marker attributes in a markers grid. Appellant notes that these GUIs are not context sensitive, but rather are specific GUIs for the grids being displayed and edited (e.g., scripts grid, and markers grid, respectively), and are nowhere describes as changing based on the nature of an expression indicated by the user (more specifically, by hovering). Thus,

these figures do not, and cannot, disclose context sensitive GUIs, nor, more particularly, with the features of claims that are dependent from claim 6.

Thus, the cited art fails to teach or suggest all the features and limitations of claim 6 (and claims dependent therefrom), and so claim 6 (and claims dependent therefrom) is patentably distinct and non-obvious over the cited art, and thus allowable.

Claim 9

In addition to the novel limitations of claims 1, 6, and 8, the cited art fails to teach or suggest **wherein the specified format is specified via a second GUI element in the GUI**, as recited in claim 9.

Cited Figure 17, discussed above, shows specific edit fields for editing marker attributes in a markers grid, specifically, fields for time code values and transcript text, but in no way discloses a GUI element in a context sensitive GUI whereby the data format for another field in the GUI may be specified.

Thus, the cited art fails to teach or suggest all the features and limitations of claim 9, and so claim 9 is patentably distinct and non-obvious over the cited art, and thus allowable.

Claim 13

In addition to the novel limitations of claim 1, the cited art fails to teach or suggest **wherein the expression comprises a variable**, as recited in claim 13.

Cited Figure 8 (8A-8C) is directed to presentation properties and their specification via GUIs, but these properties are not variables in the source code of an executing program; rather, they are configuration parameters for specifying how a viewing engine or player plays multimedia content.

Cited Figure 13 is directed to editing a scripts grid, i.e., for editing script specification attributes. As explained above, the various grids of Deutscher are means for entering and specifying data, not program source code, and the script parameters edited thereby are not program variables, but rather, configuration parameters whereby the input data (presentation file) may be configured for processing by the viewing engine/player, including specifying script parameters governing such presentation. For example, note that [0139] states: "In operation, the presentation receives the "Script Type" and the "Script Param" at a specified time code

corresponding to a point in the timeline of the playback of the presentation. The presentation then responds accordingly by changing the presentation slide displayed, or by launching a demonstration video, and so on." This is not equivalent to variables in the source code of an executing program.

Cite Figure 15 is directed to a markers grid, and is not germane for similar reasons as Figure 13. Similarly, cited Figure 17, directed to a transcriptions grid, is also not relevant to editing variables from source code of an executing program.

Thus, the references fail to disclose these features, and so claim 13, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 14

In addition to the novel limitations of claim 1, the cited art fails to teach or suggest **wherein the expression comprises a syntactic expression comprising one or more of: one or more variables; one or more constants; one or more macros; and one or more operators**, as recited in claim 14.

As one of skill would recognize, a syntactic expression is an expression that satisfies some specified syntax. Appellant notes that a single parameter has no syntax. Cited Figures 15 and 17 (and related text) disclose GUIs for data entry grids whereby various parameters may be specified, but nowhere disclose or even hint at syntactic expressions.

Thus, the references fail to disclose these features, and so claim 14, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 15

In addition to the novel limitations of claim 1, the cited art fails to teach or suggest **wherein the execution of the program is in debugging mode**, as recited in claim 15.

The Examiner asserts that Hampapuram discloses debugging an executing program. This is incorrect. As explained above, Hampapuram discloses expanding

macros in a program at edit or compile time, i.e., *statically*, not dynamically. Nowhere does Hampapuram (or Deutscher) mention or even hint at executing a program in a debugging mode.

Thus, the references fail to disclose these features, and so claim 15, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 16

In addition to the novel limitations of claim 1, the cited art fails to teach or suggest **evaluating the expression to determine the value of the expression**, as recited in claim 16.

The Examiner asserts that Hampapuram discloses debugging an executing program. This is incorrect. As explained above, Hampapuram discloses expanding macros in a program at edit or compile time, i.e., *statically*, not dynamically. Nowhere does Hampapuram (or Deutscher) mention or even hint at executing a program in a debugging mode, nor, more particularly, evaluating an expression to determine the value of the expression during execution of the program, as per claim 16.

Thus, the references fail to disclose these features, and so claim 15, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 22

In addition to the novel limitations of claim 21, the cited art fails to teach or suggest **wherein the window is substantially just large enough to display the value of the indicated expression**, as recited in claim 22.

As explained above, Hampapuram and Deutscher fail to disclose the claimed hover invoked functionality. Moreover, the cited tooltip of Hampapuram displays the macro, the value of the expanded macro, and a sentence that contains both, e.g., "macro 'NULL' expands to:0", which clearly contains more than simply the value of the indicated macro (which is not an expression from the source code of an executing program anyway). Nor are Hampapuram's tooltips editable.

Thus, the references fail to disclose these features, and so claim 22, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 23

In addition to the novel limitations of claim 21, the cited art fails to teach or suggest **wherein the window is further operable to display the indicated expression**, and **displaying the indicated expression with the value in the window, wherein the window does not include visible boundaries demarcating the displayed expression and value, wherein the window is substantially just large enough to display the indicated expression and the value of the indicated expression**, as recited in claim 23.

As explained above, Hampapuram and Deutscher fail to disclose the claimed hover invoked functionality. Moreover, the cited tooltip of Hampapuram displays the macro, the value of the expanded macro, and a sentence that contains both, e.g., "macro 'NULL' expands to:0", which clearly contains more than simply the value of the indicated macro. Nor are Hampapuram's tooltips editable.

In other words, the cited tooltips of Hampapuram (see, e.g., Figure 4) do not display an expression (as understood in the art) and its value as claimed (e.g., "k = FF"). Moreover, the GUIs of Deutscher do not display expressions with their respective values, nor, more particularly, in a window with no visible boundaries demarcating the displayed expression and value and where the window is substantially just large enough to display the indicated expression and the value of the indicated expression.

Thus, the references fail to disclose these features, and so claim 23, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

For the foregoing reasons, it is respectfully submitted that the Examiner's rejection of claims 1 and 6-23 was erroneous, and reversal of the decision is respectfully requested.

The fee of $540.00 for filing this Appeal Brief is being paid concurrently via EFS-Web. If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Appellant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-82801/JCH.

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 2009-06-23 JCH/MSW

VIII. CLAIMS APPENDIX

The following lists claims 1 and 6-23, incorporating entered amendments, as on appeal.

1. A computer accessible memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, wherein, during execution of the program, the program instructions are executable by a processor to perform:

displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;

in response to said hovering the mouse cursor over the expression, automatically displaying a GUI element proximate to the expression, wherein the GUI element includes a value of the expression;

receiving second user input to the GUI element modifying the displayed value, thereby specifying a new value for the expression; and

setting the expression in the program to the new value in response to the second user input, wherein the program continues execution in accordance with the new value of the expression.

6. The memory medium of claim 1, wherein the GUI element is context sensitive.

7. The memory medium of claim 6, wherein the GUI element comprises a control corresponding to a data type of the expression, and wherein the data type of the expression comprises at least one of:

a string data type;

a character data type;

a numeric data type;

a Boolean data type; and

an array data type.

8. The memory medium of claim 6, wherein the GUI element is operable to display the value of the expression in a specified format;

wherein if the expression comprises integer data, the specified format comprises one or more of:

decimal;

hexadecimal;

octal;

binary; and

ASCII; and

wherein if the expression comprises single or double precision, the specified format comprises one or more of:

floating point; and

scientific notation.

9. The memory medium of claim 8, wherein the specified format is specified via a second GUI element in the GUI.

10. The memory medium of claim 1, wherein the GUI element comprises:

a first portion, operable to display the value of the expression, wherein the first portion is further operable to receive the second user input modifying the value; and

a second portion, operable to display non-editable information related to the expression.

11. The memory medium of claim 10, wherein the second portion comprises a text indicator, operable to display text.

12. The memory medium of claim 10, wherein the first portion is further operable to graphically indicate that the value is editable.

13. The memory medium of claim 1, wherein the expression comprises a variable.

14. The memory medium of claim 1, wherein the expression comprises a syntactic expression comprising one or more of:

one or more variables;

one or more constants;

one or more macros; and

one or more operators.

15. The memory medium of claim 1, wherein the execution of the program is in debugging mode.

16. The memory medium of claim 1, wherein the program instructions are further executable to perform:

evaluating the expression to determine the value of the expression.

17. The memory medium of claim 1, wherein the program instructions are further executable to perform:

dismissing the GUI element based on one or more of:

third user input, indicating dismissal of the GUI element; and

lapse of a specified time period.

18. A method for debugging a program, the method comprising:

displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;

in response to said hovering the mouse cursor over the expression, automatically displaying a value of the expression in a tooltip proximate to the expression;

receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression; and

setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression.

19. A system for debugging a program, the system comprising:

a processor; and

a memory coupled to the processor, wherein the memory medium comprises program instructions implementing a graphical user interface (GUI) for debugging the program, wherein the program instructions are executable by the processor to:

display source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

receive first user input hovering a mouse cursor over an expression in the source code during execution of the program;

in response to said hovering the mouse cursor over the expression, automatically display a value of the expression in a tooltip proximate to the expression;

receive second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression; and

set the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression.

20. A system for debugging a program, the system comprising:

means for displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

means for receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;

means for automatically displaying a value of the expression in a GUI element proximate to the expression in response to said hovering the mouse cursor over the expression;

means for receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression; and

means for setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression.

21. A memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, wherein, during execution of the program, the program instructions are executable by a processor to perform:

displaying source code for the program on a display during execution of the program, wherein the executing program was compiled from the source code;

receiving first user input hovering a mouse cursor over an expression in the source code during execution of the program;

in response to said hovering the mouse cursor over the expression, automatically displaying the value of the expression in a window proximate to the expression, wherein the window is operable to display a value of the indicated expression, wherein the window does not include window title bars or menus;

receiving second user input to the window modifying the displayed value, thereby specifying a new value for the expression; and

setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression.

22. The memory medium of claim 21, wherein the window is substantially just large enough to display the value of the indicated expression.

23. The memory medium of claim 21, wherein the window is further operable to display the indicated expression, and wherein the program instructions are further executable to perform:

displaying the indicated expression with the value in the window, wherein the window does not include visible boundaries demarcating the displayed expression and value, wherein the window is substantially just large enough to display the indicated expression and the value of the indicated expression.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.